Lazy Random Walks for Superpixel Segmentation

Jianbing Shen, *Senior Member, IEEE*, Yunfan Du, Wenguan Wang, and Xuelong Li, *Fellow, IEEE*

***Abstract*— We present a novel image superpixel segmentation approach using the proposed lazy random walk (LRW) algorithm in this paper. Our method begins with initializing the seed positions and runs the LRW algorithm on the input image to obtain the probabilities of each pixel. Then, the boundaries of initial superpixels are obtained according to the probabilities and the commute time. The initial superpixels are iteratively optimized by the new energy function, which is defined on the commute time and the texture measurement. Our LRW algorithm with self-loops has the merits of segmenting the weak boundaries and complicated texture regions very well by the new global probability maps and the commute time strategy. The performance of superpixel is improved by relocating the center positions of superpixels and dividing the large superpixels into small ones with the proposed optimization algorithm. The experimental results have demonstrated that our method achieves better performance than previous superpixel approaches.**

***Index Terms*— Lazy random walk, commute time, optimization, superpixel, texture.**

I. Introduction

SUPERPIXELS are commonly defined as contracting and grouping uniform pixels in the image, which have been widely used in many computer vision applications such as image segmentation and object recognition [10], [31]. The superpixel concept was originally presented by Ren and Malik [7] as defining the perceptually uniform regions using the normalized cuts (NCuts) algorithm. The main merit of superpixel is to provide a more natural and perceptually meaningful representation of the input image. Therefore, compared to the traditional pixel representation of the image, the superpixel representation greatly reduces the number of image primitives and improves the representative efficiency. Furthermore, it is more convenient and effective to compute the region based visual features by superpixels, which will provide the important benefits for the vision tasks such as object recognition [10].

There is a large amount of literature on automatic superpixel algorithms, for example, normalized cuts [7], mean shift algorithm [5], graph-based method [11], Turbopixels [17], SLIC superpixels [21] and optimization-based superpixels [19]. However, each superpixel method has its own advantage and drawback that may be better suited for a particular application. It is still challenging to develop a high quality superpixel algorithm, which avoids the under-segmentation and locally groups the pixels respecting the intensity boundaries. The desired properties of an ideal superpixel algorithm should not only adhere well to object boundaries of image, but also maintain the compact constrains in the complicated texture regions. In order to satisfy these desired requirements, we develop a new image superpixel segmentation method by the lazy random walk (LRW) and energy optimization algorithm to achieve better performance than the previous approaches.

Our image superpixel segmentation algorithm is based on the generalized random walk (RW) algorithms [12], [22]. However, the original RW algorithm depends on the local relationship between the pixel and its corresponding seeds with the first arrival probability. This may lead to the irregular shape of the final non-uniform superpixel results. By considering the global relationships between all the pixels and the seed points, we then develop a novel superpixel algorithm using the LRW with the compactness constraints. Our LRW algorithm with self-loops effectively solves the segmentation problem in weak boundary and complex texture regions. On the other hand, the LRW based superpixel algorithm may suffer from the sensitiveness of the initial seed positions. In order to overcome these limitations and improve the performance, we further develop a new superpixel optimization approach by introducing an energy optimization framework. Our superpixel optimization strategy is essentially a compactness constraint, which ensures the resulting superpixels to distribute uniformly with the homogeneous size by relocation and splitting mechanism. Our energy function is composed of two items, the first data item adaptively optimizes the positions of seed points to make the superpixel boundaries adhere to the object boundaries well, and the second smooth item adaptively divides the large superpixels into small ones to make the superpixels more homogeneous. According to these relocated seed positions and newly created seeds by the splitting scheme, our LRW algorithm is executed again to optimize the initial superpixels, which makes the boundaries of final superpixels adhere to

Manuscript received June 5, 2013; revised October 27, 2013 and December 20, 2013; accepted January 16, 2014. Date of publication January 27, 2014; date of current version February 18, 2014. This work was supported in part by the National Basic Research Program of China (973 Program) under Grant 2013CB328805, in part by the Key Program of NSFC Guangdong Union Foundation under Grant U1035004, in part by the National Natural Science Foundation of China under Grants 61272359 and 61125106, in part by the Program for New Century Excellent Talents in University under Grant NCET-11-0789, in part by the Shaanxi Key Innovation Team of Science and Technology under Grant 2012KCT-04, in part by the Beijing Higher Education Young Elite Teacher Project, and in part by the Specialized Fund for Joint Building Program of Beijing Municipal Education Commission. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jean-Philippe Thiran.

J. Shen, Y. Du, and W. Wang are with the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: shenjianbing@bit.edu.cn; 2120111136@bit.edu.cn; 2120131072@bit.edu.cn).

X. Li is with the Center for OPTical IMagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China (e-mail: xuelong_li@opt.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2014.2302892

object boundaries very well. Our source code and other supplementary materials will be publicly available online.[1]

II. Related Work

A plenty of papers have proposed various image superpixel methods during the last decade, and we will briefly review these works in this section. The existing superpixel approaches can be roughly classified into two categories. The first category is the algorithms that do not consider the compactness constrains during the superpixels generation procedure, such as meanshift [5], and graph based [11] algorithms. In order to avoid the superpixels crossing the object boundaries, these segmentation algorithms generally produce the superpixels by over-segmenting the image. Since these algorithms do not consider the compactness constrains, which may produce the superpixels of highly irregular shapes and sizes. The second category of superpixel algorithms considers the compactness constrains, such as normalized cuts [7], lattice cut [16], TurboPixels [17], and graph cut [19] approaches. Ren and Malik [7] proposed an image superpixel approach to segment the image into a large number of small compact and homogeneous regions by the normalized cuts. The NCuts method is very powerful in feature extraction for obtaining the regular superpixels in size and shape. However, the computational cost of NCuts superpixel approach is very high and expensive when the number of superpixels or the size of image increases greatly. Levinshtein *et al.* [17] presented an efficient TurboPixel superpixel algorithm using the level set based geometric flow evolution from the uniformly placed seeds in the image. However, it exhibited relatively poor boundary adherence because of its numerical stability issues especially with complicated textures. Veksler *et al.* [19] developed an image superpixel approach by the graph cut optimization, and the superpixels were obtained by stitching each pixel that belonged to only one of the overlapping image patches.

There are other important superpixel approaches that have been proposed to fulfill the need of increasing applications, such as the algorithms in [20], [21], [24], [27], [28], and [33]. Moore *et al.* [20] used a single graph cut method to construct an optimal solution in either the horizontal or vertical direction, which took into account both the edges and the coherence of resulting superpixel lattices. Xiang *et al.* [24] proposed a lattice-like structure of superpixel regions with uniform size by learning the eigen-images from the input image, which improved the evolution speed in the TurboPixel framework. Achanta *et al.* [21] presented a simple linear iterative clustering (SLIC) superpixel algorithm, and adopted the k-means clustering approach to generate the superpixels with relatively lower computational cost. Liu *et al.* [28] formulated the superpixel segmentation problem as an objective function on the entropy rate in the graph. The entropy rate can help to cluster the compact and homogeneous regions, which also favors the superpixels to overlap with a single object on the perceptual boundaries. Zeng *et al.* [30] presented a structure-sensitive image superpixel technique by exploiting the geodesic distance. Recently, Wang *et al.* [33] proposed an edge-weighted centroidal voronoi tessellations (EWCVTs)-based superpixel algorithm, which generated the uniform superpixels and preserved the image boundaries well.

In addition, another important related work with our paper is the RW algorithm and its applications in image processing [12], [13], [22], [23]. The RW algorithm developed by Grady [12] is a very popular and leading approach for interactive image segmentation using the foreground and background seeds by the user. The RW algorithm achieves the better interactive image segmentation performance than the graph cut based segmentation method with the same user interactions. Grady and Schwartz [13] also presented a novel isoperimetric graph partitioning approach with a small isoperimetric constant and achieved high quality segmentations by solving a linear system, which was an effective seedless version of the RW algorithm. Sinop *et al.* [14] further extended the RW algorithm by incorporating a nonparametric probability density model to locate the disconnected objects for segmentation. However, the RW algorithm for interactive image segmentation lacks a global color distribution model, which will make the segmentation result sensitive to the positions and quantities of the user-defined seeds on the image objects. In order to provide more intelligent ways to understand the intention of user inputs, Yang *et al.* [22] proposed a constrained RW algorithm to facilitate the multiple user inputs with an interactive editing framework. Their method created precise contour refinement of the segmentation results. Couprie *et al.* [29] developed a power watershed segmentation framework, which contained the RW, shortest path optimization and graph cuts. We refer the interested reader to a recent book by Grady and Polimeni [26] for more about the RW algorithm by discrete calculus.

III. Our Approach

The goal of superpixel is to over-segment the input image into small compact regions with homogenous appearance. The superpixel segmentation can be considered as a pixel labeling problem where each superpixel is assigned to a unique label. Our approach begins with placing the initialized seeds of the assigned superpixels. Then, we use the LRW algorithm to obtain the initial superpixels and their boundaries. In order to further make the superpixels more compact and their boundaries more consistent with the object boundaries in image, we develop a novel energy optimization algorithm to optimize the seed positions and split the large superpixels. Fig. 1 illustrates the workflow and gives the procedure of the proposed LRW superpixel optimization algorithm. Our superpixel approach consists of two main steps. The first step is to obtain the superpixels using the LRW algorithm with initial seed points [Fig. 1(b)]. In order to improve the superpixel performance, we optimize the initial superpixels by the new energy function in the second step. Our energy includes two items: the data item makes the superpixels more homogenous with regular size by relocating the seed positions [Fig. 1(c)], and the smooth item makes the superpixels more adhering to the texture edges by dividing the

[1] *http://cs.bit.edu.cn/shenjianbing/lrw.htm*

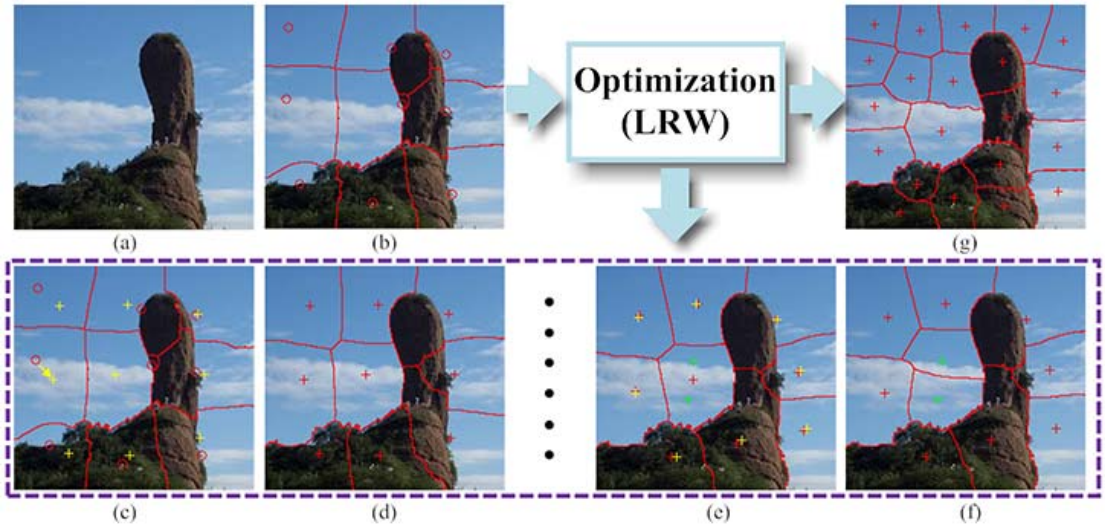


Fig. 1. The workflow of our LRW based superpixel method. (a) Input image; (b) initial superpixels by LRW and seed points (red "o"); (c) seeds relocation by superpixels optimization (yellow "+" is the relocated seeds from the original positions in (b), and yellow arrow "→" denotes the motion of some seed); (d) superpixel refinement by our LRW method with updated center positions (red "+"); (e) seeds relocation and newly created superpixels with their center positions (green "+") by superpixels optimization; (f) superpixel refinement by LRW; (g) final superpixels. Note that steps (c) to (f) (rectangle with dash lines) are performed iteratively until the final superpixels are obtained.

large irregular superpixels into small regular ones [Fig. 1(f)]. Then, our LRW algorithm is performed again to obtain the better boundaries of superpixels with these new seed positions [Fig. 1(d)]. Finally, our superpixel optimization and LRW steps are executed iteratively so as to achieve the final satisfying superpixel results [Fig. 1(g)]. In the following subsections, we will discuss in detail the LRW algorithm, LRW-based superpixel initialization and optimization algorithm.

A. Lazy Random Walk Algorithm

The RW algorithm has been used extensively for interactive image segmentation in the image processing and computer vision literatures [12], [14], [22]. The RW algorithm computes the first arrival probability that a random walk starts at one pixel first reaches one of the seeds with each label, and then that pixel is denoted as the same label with maximum probability of the corresponding seed. A random walk starts from a pixel must first arrive at the position of the pre-labeled seed, and thus it only considers the local relationship between the current pixel and its corresponding seed. This first arrival probability ignores the whole relationship between the current pixel and other seeds. As denoted by Grady [12], these limitations of the original RW method give the reason that it suffers from the weak boundary and complex texture segmentations.

In order to make full use of the global relationship between the pixel and all the seeds, we add the self-loop over the graph vertex to make the RW process lazy, which is inspired by the original LRW concept [3], [12]. However, the original LRW was initially proposed for the website data classifying and mining applications in [3], [8], and [9]. In our application, we need to further develop the original LRW algorithm to be suitable for our image superpixel segmentation application. As shown in Fig. 2, the main contribution of our LRW based superpixel algorithm is two-fold. On one hand, the self-loop [Fig. 2(b)] is added on each vertex to ensure the boundary constrain for superpixels. Since a vertex with a heavy self-loop is more likely to absorb its neighboring pixels than the one with light self-loop, which makes the vertex to absorb and capture both the weak boundary and texture information with self-loops. On the other hand, instead of starting from the pixels to the seed points as the original RW algorithm does [Fig. 2(a)], our LRW algorithm computes the commute time from the seed points to other pixels [Fig. 2(b)]. The probability maps by our LRW approach [Fig. 2(e)] give more confident separation than the ones by RW method [Fig. 2(d)]. Therefore, our LRW algorithm significantly outperforms the original RW algorithm on the test images [Fig. 2(c)] with the same background and foreground seed scribbles. We use the RW implementation[2] to produce the RW segmentation results [Fig. 2(f)]. The segmentation result by our LRW algorithm [Fig. 2(g)] achieves the better foreground objects separation than the result by RW method [Fig. 2(f)].

For describing the LRW algorithm, a graph $G = (V, E)$ is first defined on a given image $I(x_i)$, which represents a weighted graph containing a set of nodes V and edges $E \subseteq V \times V$. Then every pixel x_i is identified uniquely by a node vertex $v_i \in V$ in our undirected graph, where the degree of each vertex is computed as $d_i = \sum_j w_{ij}$ for all the edges that incident on v_i. We adopt a commonly used edge-weight computation method in many graph-based image segmentation approaches [7], [12], [23] to represent the image intensity changes. This edge-weight measures the similarity between two neighboring nodes v_i and v_j, and thus we define w_{ij} as the following Gaussian weighting function:

$$w_{ij} = exp(-\frac{||g_i - g_j||^2}{2\sigma^2}) \tag{1}$$

where g_i and g_j denote the image intensity values at two nodes

[2]http://www.cns.bu.edu/~ lgrady/random_walker_matlab_code.zip

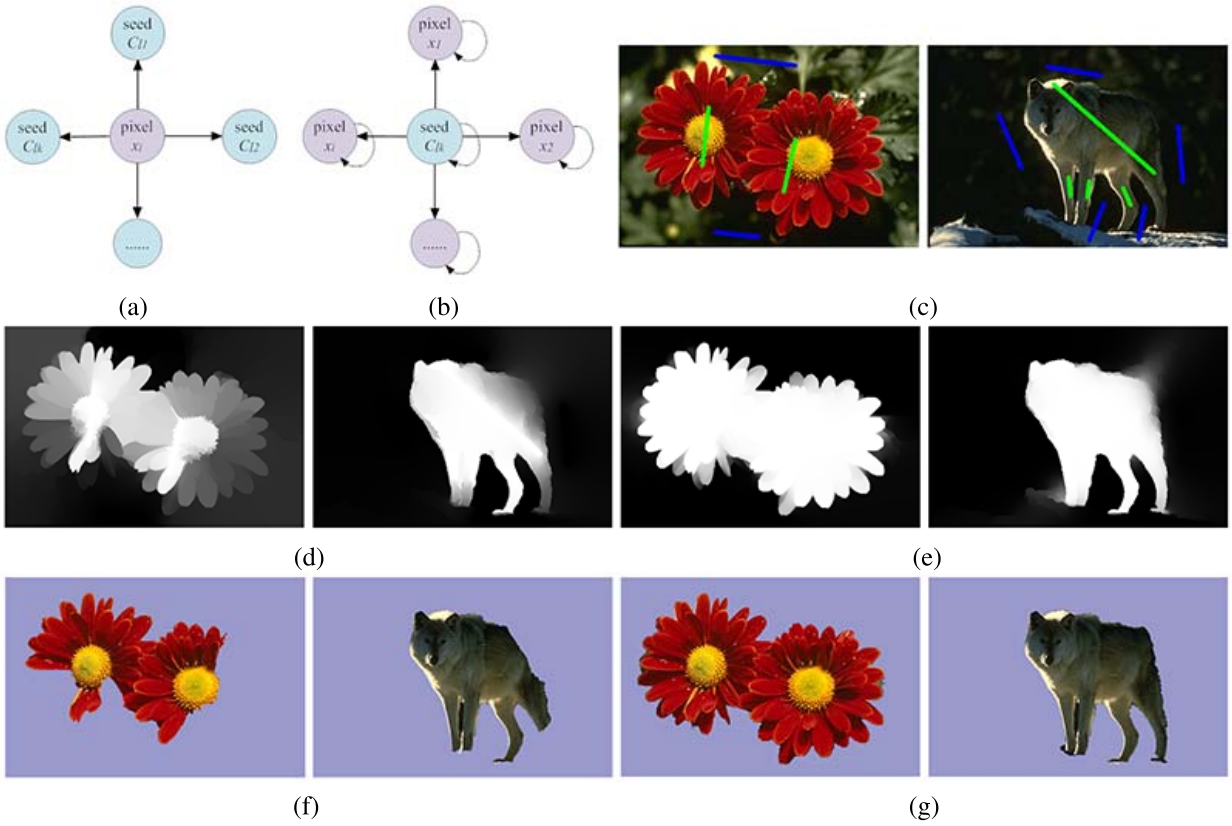


Fig. 2. Illustration the structure of RW and LRW algorithms with their comparison results. (a) Traditional RW method without self-loops; (b) our LRW algorithm with self-loops; (c) input images with user seeds (scribbles); (d) and (e) are the probability maps by RW and LRW algorithms; (f) and (g) are the segmentation results by RW and our LRW method. Image segmentation result by our LRW algorithm has the better performance than the one by classic RW method [12] with the same user scribbles (green for foreground and blue for background), especially in the leg regions of wolf and the flower parts.

v_i and v_j, and σ is the user defined parameter. The value of $2\sigma^2$ is fixed to $1/30$ in all the experiments.

As described in [12] and [32], the Gaussian function has the property of geodesic distance as the edge-weights, which works well for the proposed LRW algorithm. Comparing to the traditional RW algorithm, our LRW graph has the property that there is a non-zero likelihood that a lazy random walk remains at the same node by adding a self-loop to each vertex v_i, The new adjacency matrix is defined as

$$\mathcal{W}_{ij} = \begin{cases} 1-\alpha & \text{if i=j,} \\ \alpha \cdot w_{ij} & \text{if } i \sim j, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $i \sim j$ means node v_i and node v_j are the adjacent nodes, and α is a control parameter in the range $(0, 1)$. $\mathcal{W}$ is a sparse and symmetric banded matrix whose nonzero elements are positive.

After row-normalizing the adjacency matrix, we obtain the transition probability matrix as follows:

$$\mathcal{P}_{ij} = \begin{cases} 1-\alpha & \text{if i=j,} \\ \alpha \cdot w_{ij}/d_i & \text{if } i \sim j, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

The above equation can also be written as $\mathcal{P} = (1-\alpha)I + \alpha\mathcal{D}^{-1}\mathcal{W}$ (see **Lemma 1** in **Appendix A**) where $\mathcal{D}$ is a diagonal matrix and $\mathcal{D}_{ii}$ is the degree of the i-th vertex v_i. This means the current position v_i in our LRW algorithm will have the probability $(1-\alpha)$ to stay at the current position and the probability α to walk out along arbitrary edge. α is the sum of the weights of all the edges that incident to v_i.

According to the spectral graph theory [1], [4], the LRW algorithm converges to a unique stationary distribution π, which satisfies the following balance equation (see **Lemma 2** in **Appendix A**):

$$\pi_i = d_i / \sum_{i=1}^{N=|V|} d_i \tag{4}$$

From Equation (2), we define the following graph Laplacian matrix:

$$L_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -\alpha w_{ij} & \text{if } i \sim j, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

Equation (5) can also be written as $L = \mathcal{D} - \alpha\mathcal{W}$. We then use CT_{ij} to denote the expected quantities of steps for a lazy random walk that starts at node v_i to reach node v_j and then return to v_i. CT_{ij} is called the commute time [8], [9] between v_i and v_j, which is defined as follows:

$$CT_{ij} = \begin{cases} L_{ii}^{-1} + L_{jj}^{-1} - L_{ij}^{-1} - L_{ji}^{-1} & \text{if } i \neq j, \\ 1/\pi_i & \text{if } i = j, \end{cases} \tag{6}$$

where L^{-1} denotes the inverse of the matrix L. CT_{ij} is the corresponding Euclidean norm derived from the inner product of L_{ij}^{-1}. L^{-1} can also be considered as a Gram matrix that assigns an inner product on the vertices set.

The classic RW algorithm [1], [25] is essentially a stochastic process formed by successive summation of independent and identically distributed random variables, which usually formulates the path of a random walker with successive random steps on a discrete graph. The main reason of using the normalized Laplacian matrix is to be more consistent with the eigenvalues of adjacency matrices in spectral geometry and in stochastic process [1]. Grady [12] treated the RW probabilities algorithm as the Dirichlet problem by first decomposing the Laplacian matrix L into four sub-matrices and then converting it into a linear system. Our approach requires to solve the inverse of normalized Laplacian matrix to calculate the commute time.

From the definition of Laplacian matrix, we can see that L and L^{-1} are both the symmetric matrix and $L_{ij}^{-1} = L_{ji}^{-1}$. We then obtain the normalized Laplacian matrix $\mathcal{L} = (I - \alpha\mathcal{D}^{-1/2}\mathcal{W}\mathcal{D}^{-1/2})$ by normalizing the commute time to one that is the sum of commute time from a node to other nodes. We use $\mathcal{CT}$ to denote the normalized CT, then

$$\mathcal{CT}_{ij} = \begin{cases} 1 - \mathcal{L}_{ij}^{-1} & \text{if } i \neq j, \\ 1 & \text{if } i = j, \end{cases} \tag{7}$$

According to the property of the commute time that is inversely proportion to the probability, we now get the likelihood probabilities of label l as $f_l = \mathcal{L}_{ij}^{-1} = 1 - \mathcal{CT}_{ij}$. By defining $S = \mathcal{D}^{-1/2}\mathcal{W}\mathcal{D}^{-1/2}$, the likelihoods f_l is written as the following closed-form solution:

$$f_l = (I - \alpha S)^{-1} y_l \tag{8}$$

where I is the identity matrix and $f_l : V \to \Re$ is a $N \times 1$ vector, and the probabilities of the nodes are assigned the label l. And y_l is a $N \times 1$ column vector as $0, 0, \cdots, 0, 1, 0, \cdots, 0$ where all the elements are zero except the seed pixels as 1. Then $y_l(x_i) = 1$ if x_i is labeled with l and $y_l(x_i) = 0$ otherwise. α is chosen empirically in this work and we set $\alpha = 0.99$ through this paper to produce the sufficient good results in both the boundary adherence and superpixel homogenous.

B. LRW Based Superpixel Initialization

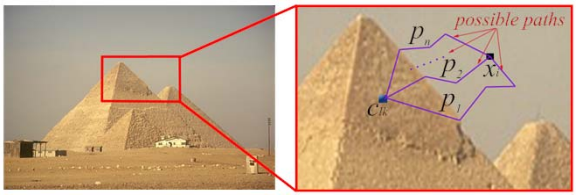
Our method begins by placing the initial superpixel seeds on the input image where we follow the similar seed initialization strategy as in [17]. Our goal is to make the superpixels to be evenly distributed over the whole image as much as possible. We first place K circular seeds in a lattice formation, and the distance between lattice neighbors is equal to $\sqrt{N/K}$ where N is the total number of pixels in the image. This strategy ensures that the superpixels will be evenly distributed on the whole image. However, this placement strategy may cause some seeds to occasionally close to a strong edge because these images are not completely uniform distribution. Thus, the initial seed position is perturbed by moving it along its gradient direction according to the seed density.

After we have finished the seed initialization stage, we then use the LRW algorithm (in Section *III.A*) to compute the boundaries of superpixels. At each step, the LRW algorithm transmits to its neighborhood with the probability which is proportional to the aforementioned edge-weight w_{ij}. The LRW algorithm will converge at a pixel x_i with the boundary likelihood probabilities $f_{l_k}(x_i)$ of superpixels as Equation (8). Finally, we obtain the labeled boundaries of superpixels from the commute time as follows:

$$R(x_i) = \underset{l_k}{argmin}\, \mathcal{CT}(c_{l_k}, x_i) = \underset{l_k}{argmax}\ f_{l_k}(x_i) \tag{9}$$

where c_{l_k} denotes the center of the l-th superpixel, and the label l_k is assigned to each pixel x_i to obtain the boundaries of superpixels.

As shown in Fig. 3, the commute time in our LRW algorithm computes the return time from the seeds to pixels, which is a proper probability measurement on the graph by considering the global information. Our LRW method can find the optimal path among all the possible LRW paths from the seed to the pixel [Fig. 3(b)]. Then the label of the seed with the minimal commute time is assigned to the corresponding pixel as the final superpixel label. Algorithm 1 gives the main steps of our LRW based superpixel initialization algorithm.



(a) (b)

Fig. 3. Illustration of computing the commute time $CT(c_{l_k}, x_i)$. Here, c_{l_k} means the k-th seed point, and x_i denotes a pixel in the image. The possible paths $\{p_1, p_2, \cdots, p_n\}$ denote all the possible LRW paths from the seed c_{l_k} to the pixel x_i, and the commute time $CT(c_{l_k}, x_i)$ denotes the average travel time of all the possible paths from c_{l_k} to x_i and return.

Algorithm 1 LRW Based Superpixel Initialization Algorithm

Input: Input image $I(x_i)$ and an integer of initial seeds K
step 1. Define an adjacency matrix $\mathcal{W} = [w_{ij}]_{M \times N}$
step 2. Construct the matrix $S = \mathcal{D}^{-1/2}\mathcal{W}\mathcal{D}^{-1/2}$
step 3. Compute $f_{l_k} = (I - \alpha S)^{-1} y_{l_k}$
step 4. Compute $R(x_i) = \underset{l_k}{argmax}\, \mathcal{CT}(c_{l_k}, x_i)$ to obtain the labels by assigning label $R(x_i)$ to each pixel x_i.
step 5. Obtain superpixels by $S_{l_k} = \{x_i | R(x_i) = l_k\}$ where $\{i = 1, \cdots, N\}$ and $\{k = 1, \cdots, K\}$
Output: the initial superpixel results S_{l_k}.

C. Superpixel Optimization

As described in the previous paragraphs, the main principle of an ideal superpixel algorithm should contain that the superpixel boundaries adhere well to image intensity boundaries and also make the superpixels to be regular with uniform size in the complicated texture regions. By considering the compactness constraints, we further improve the performance of superpixels

with the following energy optimization function:

$$E = \sum_l (Area(S_l) - Area(\bar{S}))^2 + \sum_l \widetilde{W}_x \mathcal{CT}(c_l^n, x)^2 \quad (10)$$

where the first term is the data item and the second term is the smooth item. The data item makes the texture information of image to be distributed uniformly in the superpixels, which produces more homogeneous superpixels. The smooth item makes the boundaries of superpixels to be more consistent with the object boundaries in the image. $Area(S_l)$ is the area of superpixel and $Area(\bar{S})$ defines the average area of superpixels. $\widetilde{W}_x$ is a penalty function for the superpixel label inconsistency, and we set $\widetilde{W}_x = e^{-\mathcal{CT}(c_l,x)/\beta}$ in our paper where β is a normalization factor.

When the commute time $\mathcal{CT}(c_l, x)$ between seed point c_l and pixel x is small, $\widetilde{W}_x$ will be a large value. This makes the optimized superpixel to be more compact and more homogeneous in texture regions. Zeng *et al.* [30] also adopts the similar relocation and splitting strategies to refine the over-segmentation results using the geodesic distance algorithm [15]. In contrast, our approach is intrinsically different from their method. We use LRW algorithm and the commute time to develop a novel unified energy function framework to optimize the initial superpixels. Moreover, our approach uses the local binary pattern texture measurements to define the energy items.

We choose the iterative method to solve the above energy optimization because this energy function is nonconvex. In the first step, our energy function in Equation (10) minimizes the smooth energy term to find the optimal relocation center positions of superpixels. After the first derivative with the above Equation (10) on the variable c_l^n, we obtain the new relocation center position by minimizing the data item as:

$$\begin{aligned}\frac{\partial E}{\partial c_l^n} &= 2\sum_l \widetilde{W}_x \mathcal{CT}(c_l^n, x) \nabla \mathcal{CT}(c_l^n, x) \\ &\approx 2\sum_l \widetilde{W}_x \mathcal{CT}(c_l^{n-1}, x) \frac{x - c_l^n}{||x - c_l^{n-1}||} = 0 \end{aligned} \quad (11)$$

where n represents the number of iterations and c_l^0 is the initial center positions.

For the computational convenience, we use the commute time of the $(n-1)$-th iteration to approximate the commute time of the n-th iteration, and we then use the item $\frac{x - c_l^n}{||x - c_l^{n-1}||}$ to substitute the item $\nabla \mathcal{CT}(c_l^n, x)$. Therefore, we can rewrite the above equation in a similar way to compute the center of superpixel S_l. Thus the new center relocates to:

$$c_l^n = \frac{\sum_l \widetilde{W}_x \frac{\mathcal{CT}(c_l^{n-1}, x)}{||x - c_l^{n-1}||} x}{\sum_l \widetilde{W}_x \frac{\mathcal{CT}(c_l^{n-1}, x)}{||x - c_l^{n-1}||}} \quad (12)$$

In the second step, after the first derivative with the above Equation (10) on the variable $Area(S_l)$, we get the minimizing solution of the data item as:

$$\frac{\partial E}{\partial Area(S_l)} = 2\sum_l (Area(S_l) - Area(\bar{S})) = 0 \quad (13)$$

It is obvious that the above energy has the minimal value when $Area(S_l)$ equals to $Area(\bar{S})$. The value of $Area(S_l)$ represents the texture information in the superpixel S_l. The energy reaches the minimal value when the texture is evenly distributed in each superpixel. Therefore, our energy prefers to split the large superpixels with abundant texture information into small superpixels in the optimization stage, which makes the final superpixels to be more homogeneous in the complicated texture regions.

In order to make the superpixel results adapt to the texture structure in the image, we adopt the texture feature of local binary pattern (LBP) [6] to measure the texture information. The LBP value of each pixel is defined as follows:

$$LBP_i^{q,r} = \sum_{t=0}^{q-1} s(g_t - g_i) 2^t \quad (14)$$

where q is the gray level of LBP, r is the radius of the circle around pixel i, and g means the gray value of image. We set $q = 8$ and $r = 1$ throughout this paper in our implementation. Based on the LBP texture feature, the area of superpixel is then defined as follows:

$$Area(S_l) = \sum_{i \in S_l} LBP_i, \quad Area(\bar{S}) = \frac{\sum_{i \in S_l} LBP_i}{N_{sp}} \quad (15)$$

where $Area(\bar{S})$ is the average area of superpixels, and N_{sp} is the user defined number of superpixels.

The area of superpixel is large when it contains much texture information. The parameter Th controls the number of iterations in superpixel optimization process. The larger Th is, the more iterations are required, and the better superpixels results will be achieved. The reason is that it forces the creation of more superpixels. Typically, when Th is in the range of $Th \in (1.0, 1.4)$, our method achieves the good superpixel results according to our experiments. When $Area(S_l)/Area(\bar{S}) \geq Th$, the large superpixel is divided into two small superpixels. Then the principal components analysis method is used to split the superpixel along the direction with the largest variation of commute time and superpixel shape, which is determined by $(x - c_l) \cdot s = 0$ and s denotes the corresponding eigenvector with the largest eigenvalue. This direction is the same direction of the eigenvector with the largest eigenvalue of the covariance matrix. The covariance matrix is defined as:

$$\sum_{\{x | x \in S_l, x \neq c_l\}} \frac{CT(c_l, x)^2}{||x - c_l||^2} (x - c_l)(x - c_l)^T \quad (16)$$

After the splitting procedure, we get the two new superpixels and the corresponding two new centers $c_{l_{new},1}$ and $c_{l_{new},2}$ as follows:

$$\begin{aligned} c_{l_{new},1} &= \frac{\sum_{\{(x-c_l)\cdot s > 0\}} \widetilde{W}_x \frac{CT(c_l, x)}{||x - c_l||} x}{\sum_{\{(x-c_l)\cdot s > 0\}} \widetilde{W}_x \frac{CT(c_l, x)}{||x - c_l||}} \\ c_{l_{new},2} &= \frac{\sum_{\{x | x \in S_l, (x-c_l)\cdot s < 0\}} \widetilde{W}_x \frac{CT(c_l, x)}{||x - c_l||} x}{\sum_{\{x | x \in S_l, (x-c_l)\cdot s < 0\}} \widetilde{W}_x \frac{CT(c_l, x)}{||x - c_l||}} \end{aligned} \quad (17)$$

Algorithm 2 Superpixel Optimization Algorithm

Input: Initial superpixels S_l and an integer N_{sp}
step 1. Apply Equation (12) to obtain the new c_l^n
step 2. Apply Equation (17) to get the new $c_{l_{new},1}$ and $c_{l_{new},2}$
step 3. Refine S_l by Equation (9) with c_l^n, $c_{l_{new},1}$ and $c_{l_{new},2}$
step 4. Run steps 1 to 3 iteratively until convergence
Output: the final optimized superpixel results

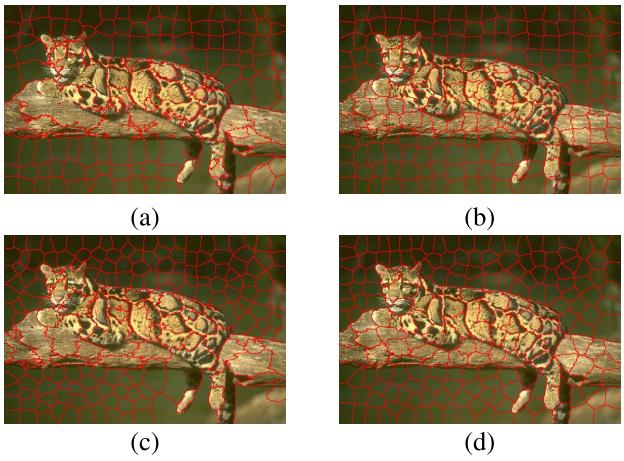


Fig. 4. Comparison results of image superpixels by RW and LRW algorithms. (a) superpixels by RW [12]; (b) superpixels by our LRW; (c) superpixels by RW and optimization; (d) superpixels by our LRW and optimization. Note that the superpixels by our LRW algorithms have the better performance such as the uniform size and good boundary adherence.

We now give a summary description of our superpixel optimization and refinement algorithm [Algorithm 2] according to the former analysis and presentation where N_{sp} denotes the number of final superpixels.

Since the LRW method considers the global relationship well between all the seeds and each pixel, it can solve the weak boundary and complex texture problems very well and obtain the superpixels with good performance. The superpixel results by our LRW algorithm [before and after optimization, Fig. 4(b) and (d)] are better than the results by the original RW algorithm [12] [before and after optimization, Fig. 4(a) and (c)]. Furthermore, our superpixel optimization algorithm plays an important role to relocate the seed positions and create the new seeds after splitting process, which further makes the regularity and boundary adherence of final superpixels by our LRW algorithm.

IV. Experimental Results

In this section, we evaluate the proposed LRW algorithm on the Berkeley segmentation database (BSD),[3] which contains three hundred test images with human segmentations as the ground-truth data [2]. The superpixel results are analyzed to explain the influence with different parameter settings. The proposed superpixel algorithm is then qualitatively tested with several representative examples in BSD benchmark. In order

[3]http://www.cs.berkeley.edu/projects/vision/grouping/segbench/

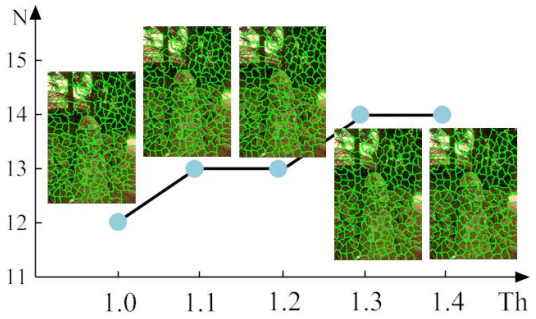


Fig. 5. Illustration of superpixel results with different values of parameter Th. The number of iterations N is plotted versus the value of threshold Th.

to obtain an objective and intuitive comparison, we quantitatively evaluate our algorithm by three different error metric measurements. We further compare our algorithm to other four well-known algorithms: Turbopixel [17], GraphCut [19], SLIC [21], and RW [12]. The superpixel results by GraphCut and SLIC in all experiments are generated by directly using the implementations from their webpages[4,5] in our comparison. The results of Turbopixel algorithm is also generated by the Matlab implementation provided by their website.[6] In order to give a relatively fair superpixel quality comparison with other algorithms, we run their programs by adjusting the parameter settings for obtaining the best superpixel performances in all our experiments. Finally, we present other comparison results between RW method and our LRW algorithm before and after optimization, and analyze the computational complexity of the representative superpixel algorithms.

A. Parameter Analysis and Results

The threshold parameter Th controls the number of iterations that are required to achieve the convergence. The number of superpixels will increase after iterations by splitting large superpixels into small ones until reaching the convergence. In the splitting process, the parameter Th helps to determine whether the current superpixel needs to be split or not. The splitting operator will be performed to the superpixel S_l if $Area(S_l)/Area(\bar{S}) > Th$ where $Area(S_l)$ is the sum of LBP values of S_l as defined in Equation(10). The purpose of splitting process is to decrease the area of large superpixel to the average area $Area(\bar{S})$, which can be viewed as the optimal size of superpixel area. Therefore, our algorithm will split the superpixel whose area is larger than the threshold area. As the value of Th increases, it will spend more iterations by relocation and splitting process to obtain the target number N_{sp} of superpixels. Fig. 5 gives the plot of the number of iterations versus the threshold value of Th where Th varies from 1.0 to 1.4. When we use the small value of Th, the required iterations of obtaining the desired superpixel results will decrease accordingly. Our algorithm achieves the satisfying superpixel results when we set Th from 1.0 to 1.4. The optimization process will take more time to converge when the value of Th is set too large. However, the superpixels will be

[4]http://www.csd.uwo.ca/faculty/olga/Code/superpixels1pt1.zip
[5]http://ivrg.epfl.ch/supplementary_material/RK_SLICSuperpixels/
[6]http://www.cs.toronto.edu/ babalex/turbopixels_code.tar.gz

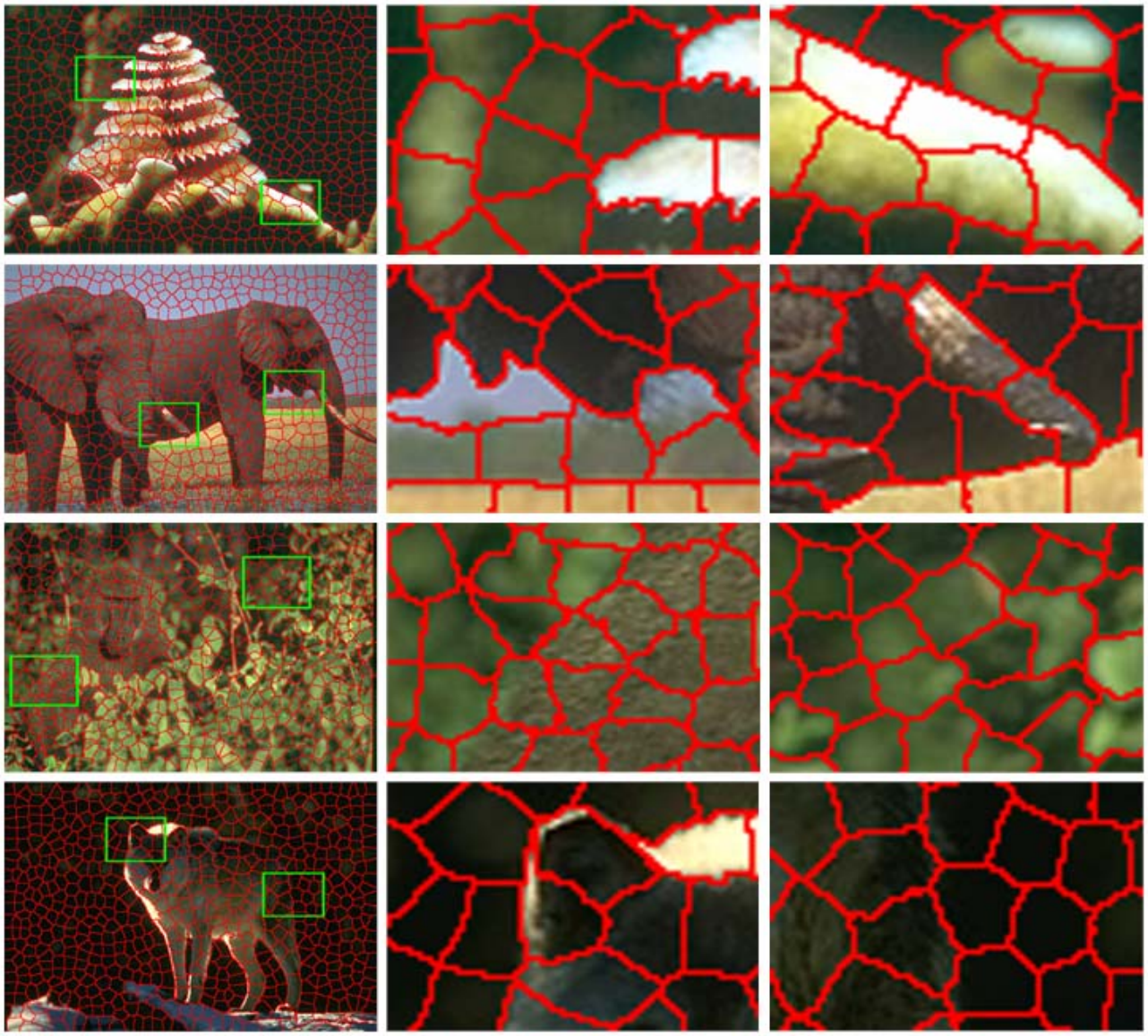


Fig. 6. Superpixel results obtained by the proposed LRW and optimization algorithm on various natural images from the BSD benchmark.

over-split and the regularity performance of them will decrease when Th is set too small ($Th < 1.0$) through our extensive experiments. We have empirically found that $Th = 1.3$ is sufficient to produce good superpixels in our experiments.

Fig. 6 gives a qualitative evaluation for the superpixel results, which are generated by the proposed LRW and optimization algorithm for the natural images from the BSD benchmark. All the examples are partitioned into roughly 400 superpixels in Fig. 6. We have observed that the superpixel boundaries by our approach adhere to the object boundaries very well, and the superpixels also possess the uniform and regular shape in size. As we have explained in Section *III.C*, our optimization strategy guarantees the boundaries of superpixels to adhere well to the object boundaries in image, such as the weak boundaries and complex texture regions in Fig. 6 (third and bottom rows). Furthermore, our superpixel optimization algorithm makes the size of superpixel as regular and uniform as possible. However, the superpixels inside the texture region will be further optimized to fit the boundaries of texture patterns. As shown in Fig. 6 (first row), the shape of superpixels in coral region is optimized to the quadrilaterals, which describes the striped texture pattern nicely.

B. Quantitative Comparison With Other Algorithms

There are three commonly used evaluation measures to evaluate the performance of superpixel algorithms. These measures include the under segmentation error (UE), the boundary recall (BR), and the achievable segmentation accuracy (ASA) [17], [21], [33]. In order to quantitatively compare our superpixel algorithm to the existing algorithms, we adopt these three metric measures to evaluate our algorithm. We use $\mathcal{GT} = \{g_1, g_2, \cdots, g_K\}$ to denote the segmentation of the ground truth image, and use $\mathcal{SG} = \{s_1, s_2, \cdots, s_L\}$ to represent the results by the superpixel algorithms.

1) Under-Segmentation Error: A good superpixel algorithm should try to avoid the under-segmentation areas in the

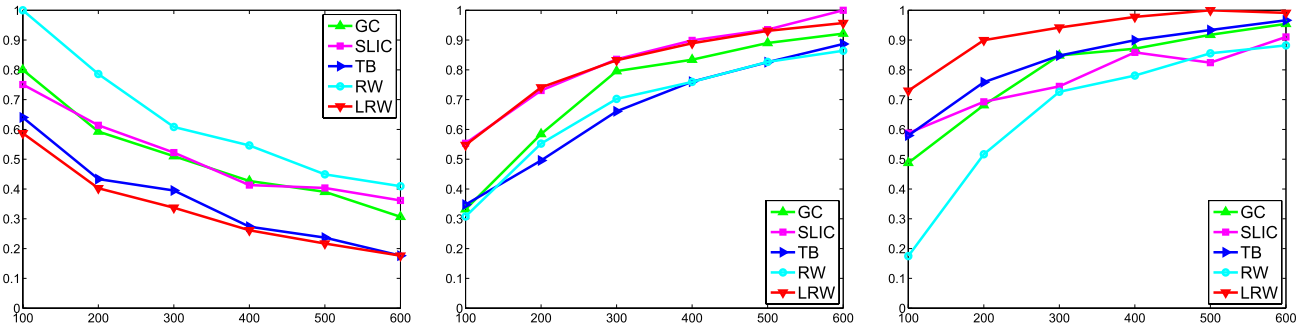


Fig. 7. Performance evaluation: (a) the curves of UE; (b) the curves of BR; (c) the curves of ASA. Our algorithm performs better than the other four algorithms (GC, SLIC, TB and RW) in both the UE and ASA measurements from lower superpixel densities to higher superpixel densities. Note that we abbreviate GraphCut superpixel [19] (GC) and Turbopixel superpixel [17] algorithm (TB).

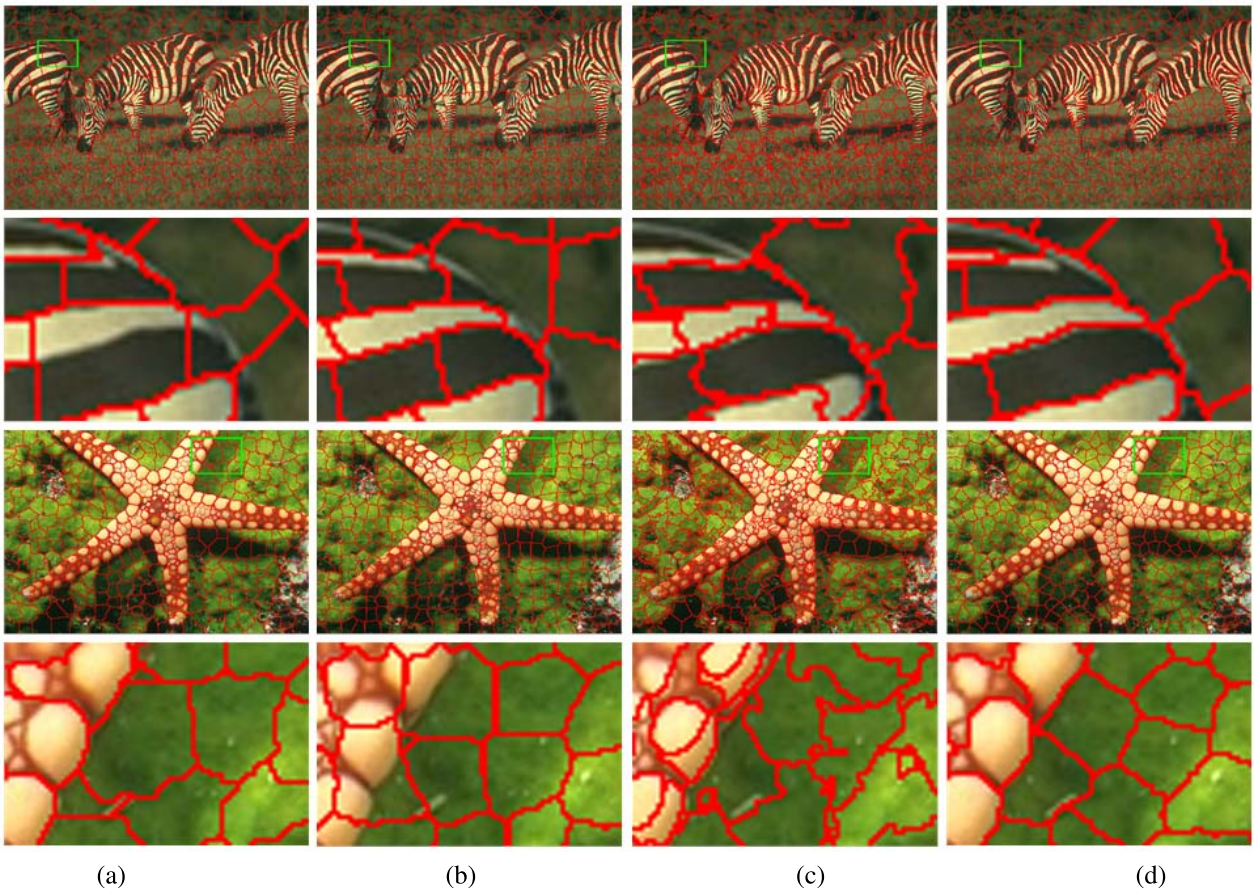


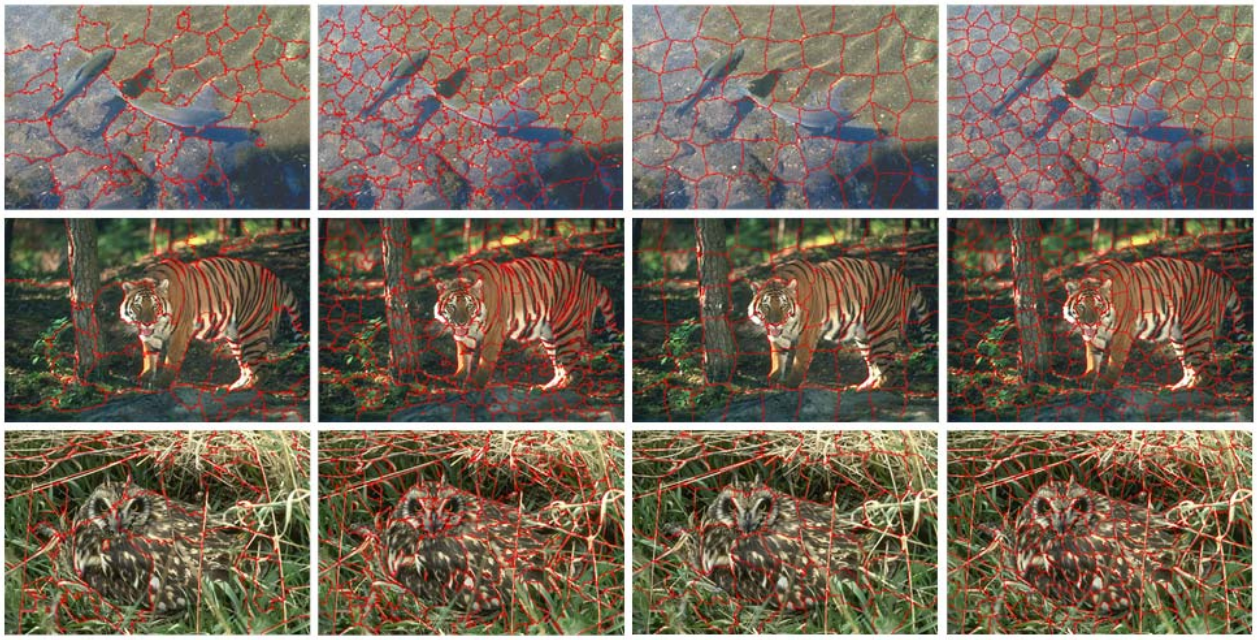
Fig. 8. Visual comparison between our algorithm and the other three well-known algorithms. (a) Results by GraphCut algorithm [19]; (b) results by TurboPixel algorithm [17]; (c) results by SLIC algorithm [21]; (d) our superpixel results.

segmentation results. In other words, we need to make sure that a superpixel only overlaps one object. This evaluation measurement checks the deducting area by the superpixel that overlaps the given ground-truth segmentation, which is calculated by the following equation [21]:

$$UE = \frac{\sum_i \sum_{s_j || s_j \cap g_i| > 0} |s_j - g_i|}{\sum_i g_i} \tag{18}$$

Equation (18) defines the UE of an image by averaging this quantity over all the ground truth segmentations. Fig. 7(a) gives the under-segmentation error of superpixel results produced by Turbopixel [17], GraphCut [19], SLIC [21] and RW [12] algorithms. All of these plots are produced by averaging the three hundred images from the BSD benchmark. As shown in Fig. 7(a), the plot of UE shows that the performance of our LRW algorithm achieves better performance than the other four algorithms. The reason is that both the GraphCut and SLIC algorithms lack a compact constraint, which may lead to the superpixel to grow from the highly irregular boundaries especially in the complicated texture regions.

2) Boundary Recall: Precise boundary is an important metric for measuring the performance of superpixel algorithms by



(a) (b) (c) (d)

Fig. 9. Comparisons between RW and LRW algorithms before and after the superpixel optimization process. (a) and (c) are the initial superpixels results using RW and our LRW algorithm (without optimization), respectively; (b) and (d) are the final superpixels results after optimization by RW and our LRW algorithm, respectively.

considering the boundary adherence. A superpixel algorithm with good ability to adhere well to the object boundaries would improve the segmentation performance. Boundary recall measurement computes the ratio of the ground truth boundaries that fall within the nearest superpixel boundaries. We use a standard measure of boundary recall [19], [21], which is defined as follows:

$$BR = \frac{\sum_{i \in \mathcal{GT}_b} logical(min_{j \in \mathcal{SG}_b} \|x_i - x_j\| < 2)}{\mathcal{GT}_b} \quad (19)$$

where $\mathcal{GT}_b$ and $\mathcal{SG}_b$ denote the boundary results obtained from the ground truth and the superpixel results, respectively.

For the boundary recall measurement, our superpixel method outperforms the GraphCut, Turbopixel and RW algorithms from the lower superpixel densities to the higher superpixel densities as shown in Fig. 7(b). The SLIC algorithm does not constrain its superpixel to be compact and uniform, which helps to capture the good boundaries of irregular regions. Our algorithm achieves the comparable performance of BR with the SLIC algorithm. However, our LRW superpixel method considers the compact constrain well and achieves the better performance of both UE and ASA measurements than the SLIC method.

3) Achievable Segmentation Accuracy: This metric measures whether the objects in the image are correctly recognized, which is also used in [28]. In other words, ASA computes the highest achievable accuracy by labeling each superpixel with the label of ground truth segmentation that has the biggest overlap area. Then the metric is defined as:

$$ASA = \frac{\sum_j \underset{i}{argmax} |s_j \cap g_i|}{\sum_i |g_i|} \quad (20)$$

Our algorithm outperforms the other four algorithms in Fig. 7(c), while the other four algorithms present the similar ASA performance. Our LRW superpixel approach generates the most correct overlap regions that share the same label between the superpixel results and the ground truth segmentations.

C. Comparison With Some Well-Known Approaches

Fig. 8 illustrates the comparison results of superpixel segmentations of different natural images between our LRW algorithm and the other well-known algorithms [17], [19], [21]. We can clearly see from Fig. 8 that our superpixel results have the uniform size in complex texture regions. Since we use LBP texture measurement to compute the energy function, our algorithm finds the accurate object boundaries in complex texture regions and simultaneously keeps the superpixels homogeneous very well. The superpixels by GraphCut algorithm adhere well to the important boundaries, but their method fails to keep superpixel homogeneous in size [Fig. 8(a)]. Because the GraphCut method only sets the upper bound of superpixel size, which leads to the generated superpixel with different size [Fig. 8(a)]. The superpixels by Turbopixel algorithm are homogeneous and regular in size, but their boundaries do not fit to the object edges well in the image [Fig. 8(b)]. The boundaries of superpixels by SLIC algorithm fit the object boundaries well in image, but the shapes of superpixels are very irregular such as the small narrow superpixels in the complex texture regions [Fig. 8(c)]. However, the main purpose of the SLIC algorithm is to provide a fast superpixel implementation by employing the k-means clustering method, which may sacrifice some performance such as the uniform size of superpixels. In contrast, our

LRW algorithm exhibits the advantages in both preserving the uniform size and boundary adherence of superpixels very well, especially the quadrilateral shape of superpixels in zebra regions in the complicated texture regions [Fig. 8(d)].

D. More Discussions

In general, the good superpixel algorithms can improve the performance of the image segmentation and computer vision applications where the superpixels are usually employed as a pre-processing step. Two important issues that we need to consider in designing the high quality superpixel algorithms are the boundary adherence and the uniform size of superpixels. In order to fully satisfy these two requirements, we first employ the proposed LRW algorithm to produce the initial superpixels and then improve them by minimizing an energy function. As shown in Fig. 9, we give the comparison results before and after the superpixel optimization stage between RW and our LRW algorithm where the input images contain both the weak boundaries and complicated texture regions. It is obvious that the performance of superpixels is improved after optimization, especially in both preserving the regular shape and adhering to the intensity boundaries well. Our LRW algorithm performs well in the image containing the complex texture regions [Fig. 9(d)]. However, the superpixel results by the RW algorithm before and after optimization exhibit the limitations such as irregular size and small broken superpixels where the boundaries adherence is also not satisfying [Fig. 9(a) and (c)]. Our termination conditions are either the changes of energy values between two successive iterations or the user defined number of superpixels.

The proposed LRW algorithm will degenerate to the classic RW algorithm when α is set to one in Equations (2) and (3). We now further analyze the computational complexity of our algorithm. The complexity of both SLIC and TurboPixel algorithm is approximately $O(N)$ where N is the total number of pixels in the image, and the complexity of GraphCut algorithm is about $O(N^2)$. We use the symmetric and highly sparse Laplacian graph to compute the LRW algorithm by solving the large sparse linear system. The complexity of our full superpixel algorithm is about $O(nN^2)$ where n is the number of the iterations. We refer the reader to [18] for more details about the algorithm complexity analysis.

V. Conclusion

We have presented a novel image superpixel approach using the LRW and energy optimization algorithm in this paper. Our method first runs the LRW algorithm to obtain the initial superpixel results by placing the seed positions on input image. Then we further optimize the labels of superpixels to improve the regularity and boundary adherence performance by relocating the center positions of superpixels and dividing the large superpixels into small uniform ones in an energy optimization framework. The experimental results have demonstrated that our superpixel algorithm achieves better performance than the previous well-known superpixel approaches. Our algorithm is capable of obtaining the good boundary adherence in the complicated texture and weak boundary regions, and the proposed optimization strategy significantly improves the quality of superpixels.

Appendix A

Lemma 1. The transition probability matrix of our LRW algorithm is formulated as $\mathcal{P}_{ij} = (1-\alpha)I + \alpha\mathcal{D}^{-1}\mathcal{W}_{ij}$.

Proof. The traditional RW method is determined by the following transition probability matrix:

$$P_{ij} = D^{-1}W_{ij} \tag{21}$$

$$W_{ij} = \begin{cases} w_{ij} & \text{if } i \sim j, \\ 0 & \text{otherwise}, \end{cases} \tag{22}$$

where W_{ij} is an adjacency matrix, and its element w_{ij} denotes the weight of edges connecting nodes v_i and v_j with $w_{ii} = 0$.

Then the transition probability from v_i to v_j is calculated as follows:

$$P_{ij} = \frac{w_{ij}}{d_i} \tag{23}$$

In contrast, the new transition probability matrix $\mathcal{P}_{ij}$ of our LRW is defined by a new adjacency matrix $\mathcal{W}$ on the graph. The difference between $\mathcal{W}$ and traditional adjacency matrix W is that we set $w_{ii} = \alpha$ in our LRW with self-loops and $w_{ii} = 0$ in original RW.

$$\mathcal{W}_{ij} = \begin{cases} \eta & \text{if } i = j, \\ w_{ij} & \text{if } i \sim j, \\ 0 & \text{otherwise}, \end{cases} \tag{24}$$

From the above equation, the new diagonal matrix $\mathcal{D}$ is defined with the new degree $d_i^{'} = d_i + \eta$. Then the new transition probability of our LRW algorithm is computed as:

$$\mathcal{P}_{ij} = \begin{cases} \frac{\eta}{d_i^{'}} & \text{if } i = j, \\ \frac{w_{ij}}{d_i^{'}} & \text{if } i \sim j \end{cases} \tag{25}$$

Let $\alpha = \frac{d_i}{d_i^{'}}$ be the probability of transiting from the current node to other nodes, while $(1-\alpha)$ is the probability to stay at the current node. Therefore, we obtain the new transition probability matrix $\mathcal{P}$ as follows:

$$\mathcal{P} = (1-\alpha)I + \alpha\mathcal{D}^{-1}\mathcal{W} \tag{26}$$

Lemma 2. There exists a unique probability distribution $\pi = \{\pi_1, \pi_2, \cdots, \pi_N\}$ for the LRW algorithm that satisfies the balance equation $\pi_i = d_i / \sum_{i=1}^{N=|V|} d_i$.

Proof. Since LRW is guaranteed to converge to a stationary distribution. Then the LRW algorithm on graph G satisfies the following distribution:

$$\begin{aligned} \mathbf{1}\mathcal{D}\mathcal{P} &= \mathbf{1}\mathcal{D}((1-\alpha)I + \alpha\mathcal{D}^{-1}\mathcal{W}) \\ &= (1-\alpha)\mathbf{1}\mathcal{D} + \alpha\mathbf{1}\mathcal{D}\mathcal{D}^{-1}\mathcal{W} \\ &= (1-\alpha)\mathbf{1}\mathcal{D} + \alpha\mathbf{1}\mathcal{W} \\ &= (1-\alpha)\mathbf{1}\mathcal{D} + \alpha\mathcal{D} = \mathbf{1}\mathcal{D} \end{aligned} \tag{27}$$

where $\mathbf{1}$ denotes a $1 \times N$ vector with all the elements equal to 1. Therefore the stationary probability distribution for any initial distribution is obtained as:

$$\pi = \mathbf{1}\mathcal{D} / \sum_i \mathcal{D}_{ii} \tag{28}$$

Let $d_i = \mathcal{D}_{ii}$, we obtain the discrete representation of the above equation as follows:

$$\pi_i = d_i / \sum_{i}^{N=|V|} d_i \tag{29}$$

REFERENCES

[1] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI, USA: Amer. Math. Soc., 1997.

[2] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE ICCV*, Vancouver, BC, Canada, Jul. 2001, pp. 416–423.

[3] D. Aldous and J. Fill. (2002). *Reversible Markov Chains and Random Walks on Graphs* [Online]. Available: http://stat-www.berkeley.edu/users/aldous/RWG/book.html

[4] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. NIPS*, 2002, pp. 849–856.

[5] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[6] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray scale and rotation invariant texture analysis with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[7] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. 9th IEEE ICCV*, Oct. 2003, pp. 10–17.

[8] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *Proc. 21st ICML*, 2004, pp. 1–9.

[9] D. Zhou and B. Schölkopf, "Learning from labeled and unlabeled data using random walks," in *Proc. DAGM*, 2004, pp. 237–244.

[10] G. Mori, X. Ren, A. A. Efros, and J. Malik, "Recovering human body configurations: Combining segmentation and recognition," in *Proc. IEEE CVPR*, Jul. 2004, pp. 326–333.

[11] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, 2004.

[12] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach Intell.*, vol. 28, no. 11, pp. 1768–1783, Nov. 2006.

[13] L. Grady and E. Schwartz, "Isoperimetric graph partitioning for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 469–475, Mar. 2006.

[14] A. K. Sinop and L. Grady, "A seeded image segmentation framework unifying graph cuts and random walks which yields a new algorithm," in *Proc. IEEE ICCV*, Oct. 2007, pp. 1–8.

[15] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," in *Proc. IEEE 11th ICCV*, Oct.2007, pp. 1–8.

[16] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *Proc. IEEE CVPR*, Jun. 2008, pp. 1–8.

[17] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2290–2297, Dec. 2009.

[18] D. S. Watkins, *Fundamentals of Matrix Computations*, 3rd ed. New York, NY, USA: Wiley, 2010.

[19] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *Proc. ECCV*, 2010, pp. 211–224.

[20] A. Moore, S. Prince, and J. Warrel, "Lattice cut—Constructing superpixels using layer constraints," in *Proc. IEEE CVPR*, Jun. 2010, pp. 2117–2124.

[21] s R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fsua, and S. Sässtrunk, "SLIC superpixels," EPFL, Lausanne, Switzerland, Tech. Rep. 149300, 2010.

[22] W. Yang, J. Cai, J. Zheng, and J. Luo, "User-friendly interactive image segmentation through unified combinatorial user inputs," *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2470–2479, Sep. 2010.

[23] V. Gopalakrishnan, Y. Hu, and D. Rajan, "Random walks on graphs for salient object detection in images," *IEEE Trans. Image Process.*, vol. 19, no. 12, pp. 3232–3242, Dec. 2010.

[24] S. Xiang, C. Pan, F. Nie, and C. Zhang, "TurboPixel segmentation using eigen-images," *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 3024–3034, Nov. 2010.

[25] G. F. Lawler and V. Limic, *PRandom Walk: A Modern Introduction*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[26] L. Grady and J. R. Polimeni, *Discrete Calculus: Applied Analysis on Graphs for Computational Science*. New York, NY, USA: Springer-Verlag, 2010.

[27] Y. Zhang, R. I. Hartley, J. Mashford, and S. Burn, "Superpixels via pseudo-Boolean optimization," in *Proc. IEEE ICCV*, Nov. 2011, pp. 1387–1394.

[28] M. Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Proc. IEEE CVPR*, Jun. 2011, pp. 2097–2104.

[29] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 7, pp. 1384–1399, Jul. 2011.

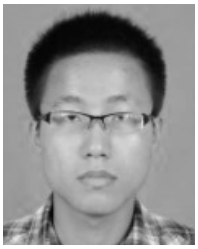
[30] G. Zeng, P. Wang, J. Wang, R. Gan, and H. Zha, "Structure-sensitive superpixels via geodesic distance," in *Proc. IEEE ICCV*, Nov. 2011, pp. 447–454.

[31] Z. Li, X. M. Wu, and S. F. Chang, "Segmentation using superpixels: A bipartite graph partitioning approach," in *Proc. IEEE CVPR*, Jun. 2012, pp. 789–796.

[32] T. Pätz and T. Preusser, "Segmentation of stochastic images with a stochastic random walker method," *IEEE Trans. Image Process.*, vol. 21, no. 5, pp. 2424–2433, May 2012.

[33] J. Wang and X. Wang, "VCells: Simple and efficient superpixels using edge-weighted centroidal Voronoi tessellations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1241–1247, Jun. 2012.

Jianbing Shen (M'11–SM'12) is currently a Full Professor with the School of Computer Science, Beijing Institute of Technology, Beijing, China. He has published more than 40 refereed papers in journals and conference proceedings. His current research interests include computer vision and computer graphics.



Yunfan Du is currently pursuing the M.S. degree with the School of Computer Science, Beijing Institute of Technology, Beijing, China. His current research interests include image segmentation using random walks.

Wenguan Wang is currently pursuing the M.S. degree with the School of Computer Science, Beijing Institute of Technology, Beijing, China. His current research interests include image cosegmentation.

Xuelong Li (M'02–SM'07–F'12) is a Full Professor with the Center for Optical Imagery Analysis and Learning, State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an, China.